

UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA
SAN FRANCISCO DIVISION

NETWORK APPLIANCE, INC.,
Plaintiff-Counterclaim Defendant,
v.
SUN MICROSYSTEMS, INC.,
Defendant-Counterclaim Plaintiff.

CASE NO. C-07-06053-EDL

**SUPPLEMENTAL DECLARATION OF
DR. SCOTT BRANDT IN SUPPORT OF
SUN MICROSYSTEMS, INC.'S REPLY
CLAIM CONSTRUCTION BRIEF**

I, Scott Brandt, Ph.D., declare as follows:

I. '292 PATENT

A. "non-volatile storage means"

1. The term "non-volatile storage" describes a function, and not a specific physical structure or class of structures that performs the function.

2. Although many structures are capable of performing the function of non-volatile storage, including those discussed in my previous declaration, in NetApp's brief, and in the declaration of Dr. Ganger, no structure is mentioned in the relevant claim(s) to explain how this function is accomplished.

3. It has been explained to me that in these circumstances, one must look to the specification of the patent to determine the intended structure. Based on my reading of the patent,

1 it is my opinion that the intended structure is “one or more disks with a block-based format (i.e.,
2 4KB blocks that have no fragments), where the disk storage blocks are the same as the data
3 blocks of the file system”

4 4. Both NetApp and Dr. Ganger state that the storage means is a passive target of
5 “writing and storing.” (*See, e.g.,* Ganger Declaration, paragraph 11.) That is incorrect. Hard
6 drives, the intended targets of the writing and storing, are active components of a storage system.
7 They execute firmware, actively read and write data to their internal storage, and perform other
8 operations independent of the requests from the host computer.

9 5. NetApp’s definition of the term “non-volatile storage means” is too broad to be
10 correct. It includes types of non-volatile data storage—such as EPROM—that are clearly beyond
11 what is intended by the patent’s authors. Dr. Ganger admits as much in his declaration (Paragraph
12 17), stating,

13 I do agree that the term “non-volatile storage” should be understood
14 broadly to include, in addition to disks, other components on which
15 file systems may be maintained, such as flash memory drives, disk
16 arrays, battery backed RAM devices, and the like. But, I believe
17 Sun and Dr. Brandt go too far in suggesting that relevant “non-
18 volatile storage” includes things like “paper” and “film” simply
19 because they retain data in the absence of power. One of ordinary
20 skill in the art would understand that, in the context of the patent,
21 which focuses on maintaining consistent states and read-only copies
22 of active file systems, the term “non-volatile storage” does not refer
23 to things like “paper” and “film,” which are not used for
24 maintaining active (computer) file systems.

25 Thus, by Dr. Ganger’s admission, paper-based non-volatile computer storage devices such as
26 paper tape readers and punch-card systems—both of which clearly fit NetApp’s proposed
27 definition—are not referred to by term “non-volatile storage” as it is used in the patent and
28 therefore “non-volatile storage means” cannot mean “A storage device that can retain information
in the absence of power,” as NetApp argues.

6. In Section A.1.c of its brief (NetApp Responsive Brief), NetApp argues that term
“non-volatile storage” generally refers to a hard disk drive. This is inconsistent with NetApp’s
definition of “non-volatile storage means” as “storage device[s] that can retain information in the
absence of power,” as that definition includes many other types of many other types of “storage

1 device[s] that can retain information in the absence of power,” including those that are not
2 intended by the patent’s authors (per Ganger’s Declaration, paragraph 17). The argument is,
3 however, consistent with Sun’s definition.

4 7. Fundamental to the operation of WAFL is "a disk format system that is block
5 based (i.e. 4 KB blocks that have no fragments)." ('292 patent, col. 5:48-53). WAFL formats the
6 disk so that it is prepared to store 4KB storage blocks corresponding to the 4KB data blocks.
7 NetApp criticizes Sun's construction because it does not identify all the requirements of WAFL—
8 such as WAFL inodes and directories—as corresponding structure. However, inodes and
9 directories are not inherently structural aspects of the storage means and are, therefore, not
10 included in the minimally necessary structure. In contrast, the block-based disk format is
11 inherently structural because it describes how the underlying disk stores the data.

12 8. In my opinion, if one of ordinary skill in the art were to read the patent, Sun’s
13 definition is the understanding that they would take away from it, that “non-volatile storage
14 means” refers to “one or more disks with a block-based format (i.e., 4KB blocks that have no
15 fragments), where the disk storage blocks are the same as the data blocks of the file system.”

16 **B. “metadata for successive states of said file system”**

17 9. As I discussed in my declaration at paragraphs 68-76, from the perspective of one
18 of ordinary skill in the art, the language of claim 8 reciting “meta-data for successive states of the
19 file system” means “a block map file recording snapshots of the file system.”

20 10. Claim 8 recites a single, three-step process for creating a plurality of snapshots,
21 recited as “read-only copies of the file system.” ('292 patent, claim 8, col. 26:1-2.) The first step
22 of “storing meta-data for successive states of the file system” enables the creation of the snapshot.
23 ('292 patent, col. 18:21-23.) The second and third steps are directed to the creation of the
24 snapshot. ('292 patent, col. 26:5-15.)

25 11. The block map is the unique element described within the '292 patent for “storing
26 meta-data for successive states of the file system.” It is the structure that uniquely stores metadata
27 for successive snapshots.
28

1 12. NetApp construes the term “metadata for successive states of a file system” as
2 meaning “Information that describes successive ‘states of a file system’.” NetApp’s construction
3 replaces the word “metadata” with the broader term “information.” The word “information” is
4 generic and is not synonymous with the narrower term “metadata” in any file system context. To
5 substitute “information” for “metadata” causes the use of the term “metadata” to have no bearing
6 on the scope of the claim.

7 13. The term “information” broadly includes many things that are not “metadata,”
8 including file data itself. Dr. Ganger’s explanation indicates that “metadata,” even in the broad
9 sense he adopts, should not include the “corresponding data.” (Ganger paragraph 29). Dr.
10 Ganger’s explanation is inconsistent with the specification which states, for example, that the
11 “[f]ile system information structure 1510 is not a meta-data file...” (’292 patent, col. 10:60-62).
12 NetApp’s construction fails to define the phrase as it would be understood by one of ordinary
13 skill in the art.

14 14. The patent is very clear about what is meant by the term “metadata”, saying,
15 “WAFL uses files to store meta-data that describes the layout of the file system. WAFL meta-data
16 files include: an inode file, a block map (blkmap) file, and an inode map (inomap) file.” (’292
17 patent, col. 5:53-56). Of these, the only file that describes successive states of the file system is
18 the block map file. The other two, the inode file and the inode map, are unique to a specific state
19 of the file system.

20 15. Dr. Ganger states that “‘metadata’ is commonly understood to include many
21 structures that contain information about the corresponding data.” (Ganger paragraph 29). This
22 observation is misplaced because it ignores the language of the Claim, which identifies the term
23 being defined as “metadata for successive states of said file system.” Specifically, none of the
24 examples provided by Dr. Ganger, such as “an inode file, a root inode, an inode map file, inode
25 tables, directories, bitmaps, and indirect block” are associated with “storing metadata for
26 successive states of the file system.” The phrase itself significantly identifies the metadata that is
27 required, i.e., the metadata for successive states of the file system.

28 16. The word “successive” informs and confirms that “states of the file system” refers

1 to the current state of the file system and past states of the file system that may be recorded
2 snapshots.

3 17. “State of a file system” means “a set of storage blocks for a file system that
4 includes all blocks required for the data and structure of the file system.” (Brandt decl., paragraph
5 131). A person of ordinary skill in the art recognizes that the plural “states” refers to more than
6 one set of storage blocks for the file system that include all blocks required for the data and
7 structure of the file system. In addition to indicating more than one state of the file system, the
8 claim language includes the word “successive,” further indicating that multiple states are
9 intended.

10 18. The ’292 patent is clear in describing the succession from a past state of the file
11 system to a current state of the file system in Figure 18C. The ’292 patent explains that a snapshot
12 is created as the past state of the file system and subsequently the file system enters the current
13 state. (’292 patent col. 19:20-23). The ’292 patent refers to the current state of the file system as
14 the most recent consistency point written to disk. (’292 patent, col. 11:60-12:1, col. 14:42-44, col.
15 17:60-64).

16 19. The block map file is copied at each successive state of the file system. (’292
17 patent col. 13:45-46.) Each snapshot will create a copy of the block map file. (’292 patent, col.
18 21:51-52). Additionally, the block map disclosed in the specification of the ’292 patent includes
19 “planes” that correspond to the active file system and each snapshot. (’292 patent, col. 10:7-15).
20 At the creation of a snapshot, the active file system bits are copied into the plane corresponding to
21 the snapshot. (’292 patent, col. 20:40-41). By copying the active file system plane, the bit map
22 file ensures that new data would not overwrite old data that was being saved in a snapshot. (’292
23 patent, col. 21:9-11). The ’292 patent consistently states that the block map file is updated by this
24 type of copying. (’292 patent, col. 13:41-44, col. 21:53-56).

25 20. Claim 8 has a broader scope than claims 9 and 10 under Sun’s construction. A
26 block map in the file system literature does not have a universally agreed upon arrangement.
27 Claim 8 discusses operation of the block map file in general.

28 21. NetApp argues that claim differentiation with Claims 9 and 10 means that claim 8

1 cannot refer specifically to the block map. I disagree. In my opinion, claims 9 and 10 describe
2 specific aspects of a block map and the ways in which the block map functions.

3 22. Claim 9 explains that the block map includes multiple usage bits per block. Claim
4 9 also has the additional requirement of “the marking of said blocks” by requiring that the
5 marking occur through placing entries in the block map. (’292 patent, col. 26:16-21). This further
6 narrows the scope of claim 8 by requiring the relationship between these claimed features.

7 23. Claim 10 is different from Claim 9 in that it further requires “multiple usage bits
8 per block” (’292 patent, col. 26:23) (as opposed to, for instance, a single multi-bit entry per
9 block). This language further limits the use and arrangement of the block map file of claim 8 by
10 designating the entries of the block map.

11 24. Claims 11-13 and 18-19 also further narrow the scope of claim 8 by requiring
12 additional structure within the scope of the claim. Sun’s construction of claim 8 is consistent with
13 the addition of structures as are recited in these claims. The additional structures work in addition
14 to the required block map file. With claim 11, the block map file and pointers are required, and
15 thus claim 11 is narrower. Similarly for claims 12, 13, 18, and 19, the “structures,” “inodes,”
16 “root structure,” and narrower “root inode” are present in addition to the block map file of claim
17 8.

18 **C. "file system information structure"**

19 25. As I discussed in my first declaration in paragraphs 77-86 and as I stated therein,
20 the phrase “file system information structure” does not have an established meaning in the file
21 system art (Brandt Declaration paragraph 82). As such, this term is defined by reference to the
22 claims and the specification, beginning with the Abstract.

23 26. NetApp’s construction of the term “file system information structure” as “a data
24 structure containing information about the layout of a file system” causes the claim language to
25 be repetitive of other claim language by causing claim 4 to read as follows: “...said first data
26 structure containing information about the layout of a file system comprising data describing a
27 layout of said file system ...”.

28 27. More generally, NetApp’s definition is too broad when viewed from the level of

1 ordinary skill in the file system art based on the '292 patent. Under NetApp's construction the
2 phrase may include the block map that is explicitly excluded from the phrase by the specification
3 ('292 patent col. 5:54-56, col. 10:60-62).

4 28. The patent support's Sun's definition of "file system information structure" as a
5 "data structure that contains the root inode of a file system in a fixed location on disk," saying:
6 "FIG. 15 is a diagram illustrating a file system information (fsinfo) structure 1510. The root inode
7 1510B of a file system is kept in a fixed location on disk so that it can be located during booting
8 of the file system. File system information structure 1510 is not a meta-data file but is part of the
9 WAFL system. The root inode 1510B is an inode referencing the inode file 1210. It is part of the
10 file system information (fsinfo) structure ..." ('292 patent, col. 10:57-64). In other words, the file
11 system information (fsinfo) structure contains the root inode and is kept at a fixed location on
12 disk.

13 29. The fsinfo structure and the root inode it contains are kept at a fixed location on
14 disk for a very important reason: the WAFL file system will not function properly if it is not.
15 Because the WAFL file system—the Write Anywhere File Layout (WAFL) file system—stores
16 all files, including the metadata files, at arbitrary locations on disk, it must store some root data
17 structure at a known location in order to enable the file system to start up after the system has
18 been powered down. One of ordinary skill in the art would understand that that data structure is
19 the file system information structure and the root inode it contains, which are stored at a fixed
20 location on disk. The patent makes this very clear and teaches no alternative.

21 30. NetApp argues that differentiation from Claim 6 requires that the fsinfo structure
22 described in claim 4 must not be defined to be at a fixed location on disk. That is incorrect.

23 31. Claim 4 describes writing an fsinfo structure at a fixed location on disk and then
24 writing a second fsinfo structure. Claim 5 is narrower than Claim 4 and recites storing copies of
25 the fsinfo structure to a first and a second location.

26 32. Claim 6 recites that both the first and second locations for the copies of the file
27 system information (fsinfo) structure are fixed and predetermined. This further limits claim 5 in at
28 least by requiring both of the copies to be stored at predetermined fixed locations.

1 33. A person of ordinary skill in the art would recognize from viewing claim 4-7
 2 together that no dependent claim from claim 4 (claiming the single file system information
 3 (fsinfo) structure) recites the requirement of storing the single copy in a fixed location. The
 4 absence of a dependent claim with that requirement is consistent with the requirement of claim 4
 5 that recites the “file system information structure” that means “a data structure that contains the
 6 root inode of a file system in a fixed location on disk.”

7 34. NetApp further argues that the file system would work just as well if the fsinfo
 8 structure and root inode were not stored at a fixed location on disk. That is incorrect. Any
 9 scenario for storing the fsinfo structure requires that it be at a fixed location, as shown in three
 10 hypothetical examples below:

11 a) The fsinfo structure and its root inode could be stored at a
 12 second fixed location, with a pointer to it stored at a first fixed
 13 location. That leaves the fsinfo structure at a fixed location,
 obviating the need for the pointer and remaining consistent with
 Sun’s definition.

14 b) The fsinfo structure and its root inode could be stored in the
 15 inode file. This results in an impossible circular situation in which
 16 one cannot locate the inode file without first locating the root inode
 and one cannot locate the root inode without first locating the inode
 file.

17 c) The fsinfo and its root inode could be stored in an arbitrary
 18 location other than the inode file, with a pointer to that location
 19 stored in a fixed location. In that case, the block map would have to
 20 be updated to reflect the use of the block or blocks to store the
 21 fsinfo structure. However, the block map file cannot be updated
 22 when the fsinfo structure is written because the patent makes clear
 23 that at the time that the fsinfo structure and its root inode are written
 24 to disk, the block map has already been written to disk, saying
 “referring to FIG. 16, a new consistency point is written 10 by first
 flushing all file system blocks to new locations on disk (including
 the blocks in meta-data files such as the inode file 1620, blkmap file
 1630, and inomap file 1640). A new root inode 1610B and 1612B
 for the file system 1670 is then written to disk.” (’292 patent, col.
 12:9-14).

25 Thus the fsinfo and its root inode must be written to a fixed location on disk.

26 35. As explained above, I disagree with NetApp’s argument that one of ordinary skill
 27 in the art would understand that the fsinfo structure and its root inode were not at a fixed location
 28 on disk.

II. '211 PATENT

A. "Pointing directly and indirectly to buffers in said memory and a second set of blocks on said storage system"

36. The term "pointing directly and indirectly to buffers in said memory and a second set of blocks of said storage system" is clear and unambiguous and one of ordinary skill in the art would understand that it means "pointing directly and indirectly to buffers in said memory and pointing directly and indirectly to a second set of blocks on said storage system."

37. The word "and" has a very specific meaning in computer science that is well understood by one of ordinary skill in the art that is explicitly distinct from the meaning of "or" and its informal equivalent "and/or". "A and B" always means "both A and B." It never means "either A or B." Thus "pointing directly and indirectly" means "pointing both directly and indirectly" or "pointing directly and pointing indirectly" while "to buffers and blocks" means "to both buffers and blocks" or "to buffers and to blocks." Thus "pointing directly and indirectly to buffers in said memory and a second set of blocks of said storage system" would be unambiguously understood to mean "pointing directly and pointing indirectly to buffers in said memory and pointing directly and pointing indirectly to a second set of blocks of said storage system" or, more concisely, "pointing directly and indirectly to buffers in said memory and pointing directly and indirectly to a second set of blocks on said storage system."

38. Some consequences of the above are that:

- a) The root inode must point directly to blocks
- b) The root inode must point indirectly to blocks
- c) The root inode must point directly to buffers
- d) The root inode must point indirectly to buffers
- e) The root inode must point to both blocks and buffers
- f) The blocks and buffers together store the second consistent state of the file system

39. The plain language of this claim term makes clear that this is what is intended and

1 these facts would have been apparent to one of ordinary skill in the art. For these reasons, I
2 disagree with Dr. Ganger's view that the fact that each individual block is not pointed to both
3 directly and indirectly somehow requires that the "and" be read as an "or." To the contrary, to
4 meet this claim limitation, even a minimal file system must have at least one block pointed to
5 directly and at least one other block pointed to indirectly as well as at least one buffer pointed to
6 directly and one pointed to indirectly. Any other way of reading the claim language is
7 inconsistent with its plain meaning and the knowledge of one of ordinary skill.

8 40. The surrounding claim language confirms this understanding. The claims require
9 that the second consistent state referenced by the incore root inode be stored in "said buffers and
10 said second set of blocks." ('211 patent, col. 24:2-6). This means that both blocks and buffers
11 must be present to together comprise the second consistent state. Dr. Ganger recognizes that
12 these buffers and blocks together comprise one group, but somehow concludes that the preceding
13 "pointing directly and indirectly" language does not need to apply to both buffers and blocks. I
14 disagree. If both buffers and blocks are required to store the second consistent state, the other
15 "ands" in the claim must be interpreted in the conjunctive for the claim to make sense.

16 41. Dr. Ganger provides two examples that he believes show situations where the root
17 inode would not point directly and indirectly to blocks and to buffers. Neither example is
18 disclosed in the specification. Furthermore, both examples are actually counter to the teachings of
19 the specification and the claim language, as I discuss below. Dr. Ganger's first example of "not
20 pointing directly to any blocks" (Ganger Declaration, paragraph 64) is inconsistent with the
21 language of the claim. Dr. Ganger envisions a situation in which all 16 blocks pointed to directly
22 by the root inode have changed, such that "the incore root inode for the second consistent state
23 will not point directly to any blocks in the second consistent state." (Ganger Declaration,
24 paragraph 64). Claim 1 says, "...said incore root inode pointing directly and indirectly to buffers
25 in said memory and a second set of blocks on said storage system, said buffers and said second
26 set of blocks storing data and metadata for a second consistent state of said file system... ." Dr.
27 Ganger incorrectly infers from this that the incore inode cannot also point "directly" to blocks
28 from the first consistent state, even though the claim goes on to say, "... said second set of blocks

1 including at least some blocks in said first set of blocks...” and the specification makes clear that
 2 the root inode includes a copy of the on-disk inode, which points directly (and indirectly) to the
 3 blocks comprising the first consistent state (’211 patent, element 1010D in Fig. 10, col. 7:58-60).
 4 The copy of the on-disk root inode in the incore root inode continues pointing to blocks in the
 5 first consistent state even when all such blocks have undergone changes and are represented by
 6 dirty incore buffers. Therefore, consistent with the plain claim language, the incore root inode
 7 always points directly and indirectly to blocks and buffers.

8 42. Dr. Ganger also envisions a situation in which an impossibly tiny file system may
 9 not point indirectly to any buffers (Ganger Declaration, paragraph 65). This example is also
 10 incorrect, for three reasons. First, Dr. Ganger incorrectly assumes that the in-core inodes are the
 11 same size and have the same structure as the on-disk inodes. This is contradicted by the
 12 specification, which says: “The incore WAFL inode 1010 has a size of approximately 300 bytes.
 13 The on-disk inode is 128 bytes in size.” (’211 patent, col. 7:12-14). Second, a minimal inode file
 14 needs space for at least 23 inodes for the block map file, the inode map file, 20 snapshots, and a
 15 root directory (see, for example, ’211 patent, Figure 22). 23 inodes of 300 bytes each would
 16 require 6900 bytes of storage and will not fit in a single 4K buffer, as described by Dr. Ganger.
 17 Therefore, storing a file system in one buffer is not possible even under the unrealistic conditions
 18 of Dr. Ganger’s example. Third, the patent says “pointing directly and indirectly to **buffers...**”
 19 while Dr. Ganger’s example includes only a single buffer and thus cannot be what was envisioned
 20 by the claim.

21 **B. “root inode”**

22 43. NetApp construes this term to mean “An inode that points directly and/or
 23 indirectly to all the blocks in a consistent state of a ‘file system’ (as construed herein).” This
 24 definition is incorrect for two reasons. First, the claim language makes very clear that the root
 25 inode points “directly **and** indirectly.” Second, it falls short as a definition as it fails to capture a
 26 critical aspect of the root inode that is necessary for the correct operation of the file system.
 27 Namely, that it must be stored in a fixed location on disk.

28 44. One of ordinary skill in the art would look to the specification, in addition to the

1 claims, in order to understand the significance of the root inode. The specification distinguishes
2 the claimed file system from prior art file systems on the ground that the allegedly novel inode
3 file can be stored anywhere on disk, unlike prior art inode tables that were stored in a fixed
4 location. ('211 patent, col. 9:27-36). The specification makes clear that this write-anywhere
5 property of the inode file (the file that contains all file system inodes except for its own inode—
6 the root inode) necessitates that its root be in a fixed location. This is further evidenced by its
7 name, root inode, which one of ordinary skill in the art would understand to indicate that it roots,
8 or anchors, the file system by virtue of being stored in a fixed, known location.

9 45. NetApp and Dr. Ganger seem to believe that the incore and on-disk root inodes are
10 two unrelated things. In fact, the root inode is a temporary copy of the on-disk root inode stored
11 in memory. The specification teaches that incore inodes are stored in memory buffers ('211
12 patent, col. 6:55-61). The inode buffers structure shown in Figure 8 includes a copy of the on disk
13 inode 820 as well as buffer-specific elements such as flags and buffer pointers. While the root
14 inode resides in memory, the buffer pointers point directly and indirectly to other buffers,
15 including those with changed data ('211 patent, Figure 17L). At the same time, the incore copy of
16 the root inode continues to point to the same blocks as the original root inode stored on disk. The
17 incore copy of the root inode is periodically written to a fixed location on disk where it overwrites
18 the on-disk copy, synchronizing the two copies and transitioning the file system to the next
19 consistency point. Sun's construction is consistent with the operation of the root inode laid out in
20 the specification and described here. One of ordinary skill in the art would understand that the
21 claimed incore root inode is an in-memory copy of the root inode stored at a fixed location. The
22 fact that the incore copy has some additional structures and is not stored in a fixed location *in*
23 *memory* is simply not relevant to this understanding.

24 46. Dr. Ganger asserts in paragraph 79 that "One of ordinary skill in the art would
25 understand that there are other mechanisms for ensuring that the root inode for the active file
26 system can be located" and offers two examples of purported solutions for locating the root inode
27 without storing it in a fixed location. The first example is incorrect, and the second is irrelevant.

28 47. Dr. Ganger's first example says "rather than storing the root inode itself in a fixed

1 location, the file system could instead store a pointer to the root inode in a fixed location.”
2 (Ganger Declaration paragraph 79). This example has problems similar to those discussed with
3 reference to the file system information (fsinfo) structure in paragraph 34 above. If the root inode
4 is not stored in a fixed location, then storage for it must be allocated via the block map. However,
5 the specification makes clear that the block map is always written before the root inode, saying
6 “Referring to FIG. 16, a new consistency point is written by first flushing all file system blocks to
7 new locations on disk (including the blocks in meta-data files such as the inode file 1620, blkmap
8 file 1630, and inomap file 1640). A new root inode 1610B and 1612B for the file system 1670 is
9 then written to disk.” (’211 patent, col. 12:8-13). Allocating space for the root inode would thus
10 require changing the block map after it has already been written to disk for the current
11 consistency point, contradicting the teachings of the specification and potentially leading to an
12 inconsistency in the file system. This is made clear by the discussion of what the specification
13 refers to as the “single ordering constraint” of the claimed file system (’211 patent, col. 14:18-
14 28), which says in part, “The present invention, as illustrated in FIGS, 20A-20C, has a single
15 ordering constraint. The single ordering constraint is that the fsinfo block 1810 is written to disk
16 only after all the other blocks are written to disk.” Thus, one cannot update the block map to
17 write the root inode.

18 48. Dr. Ganger’s second example says that “the file system could have a set of
19 predetermined locations that might hold the root inode – to find the root inode, it would then read
20 all locations in the set to determine which actually held the root inode.” (Ganger Declaration
21 paragraph 79). In my opinion, this example is consistent with Sun’s definition, which requires
22 that the root inode be written to at least one fixed location. In fact, the specification teaches that
23 the root inode is stored in a set of fixed locations, namely two, (’211 patent, col. 11:3-5), which
24 must be scanned to determine which is the most recent in case of a file system crash. (’211 patent,
25 col. 14:54-15:5). In other words, Dr. Ganger’s second example is not an example of something
26 other than storing the root inode in a fixed location.

27 49. In its brief, NetApp confuses root inodes and snapshot inodes, saying, “In any
28 event, the specification clearly indicates—and one of ordinary skill in the art would understand—

1 that copies of the root inodes for previously saved consistency points (i.e., snapshot inodes) can
 2 be stored anywhere in the file system.” (NetApp Responsive Brief, page 29, lines 22-24). While it
 3 is true that snapshot inodes may be stored anywhere, snapshot inodes and root inodes are
 4 different, as indicated by the specification, which distinguishes the two, saying, “Each snapshot is
 5 represented by a snapshot inode that is similar to the representation of the active file system by a
 6 root inode.” (’211 patent, col. 18:3-5) and explaining how they differ (see, for example, col. 22:8-
 7 13).

8 50. In fact, Figure 22 shows that snapshot inodes reside in the inode file alongside all
 9 of the other regular inodes. The inode file is pointed to by the root inode. This means that the file
 10 system must be able to find the root inode—through having it stored in a fixed location—in order
 11 to access any of the snapshot inodes. If the root inode is lost, the snapshots are lost with it.
 12 Because snapshot inodes are located in the inode file, they are part of the claimed “metadata”
 13 pointed to directly and indirectly by the root inode. As can be seen, snapshot inodes are not
 14 themselves the root inode and are irrelevant to the proper construction of “root inode.”

15 51. The distinction between the root inode and snapshot inodes is clearly intended by
 16 the claims, which indicate that the root inode points to a consistent state of a file system, and not
 17 to a “read-only copy of a file system at a given instant when the snapshot is created” (’211 patent,
 18 col. 17:59-61). Therefore, while the root inode roots the active, evolving file system, snapshot
 19 inodes root only a read-only copy of how the file system looked in the past. The two inodes thus
 20 have a different purpose. For this reason, one of ordinary skill in the art would have understood
 21 that the properties of a snapshot inode are irrelevant to the definition of a root inode.

22
 23 **C. “consistent state”/“state of a file system”**

24 52. I understand that NetApp maintains that the term “consistency point” and the term
 25 “consistent state” should be given the same construction, “A set of blocks on disk, rooted by a
 26 root inode, that includes all the blocks required for the data and file structure of a file system.” I
 27 do not agree that these two terms are equivalent.

28 53. The term consistency point does not appear in the claims of the ’211 patent but

1 appears in claim 4 of the '292 patent, which describes a file system that transitions from one
2 consistency point to another. The specification defines consistency point as "the set of self-
3 consistent blocks on disk that is rooted by the root inode." ('211 patent, col. 4:17-19). The
4 specification further teaches that "A new consistency point occurs when the fsinfo block is
5 updated by writing a new root inode for the inode file into it. Thus, as long as the root inode is not
6 updated, the state of the file system represented on disk does not change." ('211 patent, col. 4:21-
7 25). A consistency point is therefore an on-disk state of a file system that occurs when the file
8 system on disk is rooted by the on-disk root inode. This can be seen in '292 claim 4, where the
9 file system remains at the first consistency point all the way until a new fsinfo structure
10 (containing the root inode) is written out to disk.

11 54. The term "consistent state" appears in the claims of the '211 patent. Claim 1
12 makes clear that a consistent state may include only disk blocks (the claimed "first consistent
13 state"), or may include both in-memory buffers and disk blocks (the claimed "second consistent
14 state"). This is the moment between the first and second consistency points of Claim 4 of the '292
15 patent, before the second consistency point has been written to disk. In the language of the '211
16 patent, the "on-disk root inode pointing directly and indirectly to a first set of blocks on said
17 storage system that store a first consistent state of said file system" is the "first consistency
18 point".

19 55. However, even though the in-core root inode roots a second consistent state, it
20 does not root a second consistency point because it has not yet been written to disk. In other
21 words, for the duration of the operation described in the claims of the '211 patent, the file system
22 remains at the first consistency point. Thus, the terms "consistent state" and "consistency point",
23 although related, are not synonymous and cannot be defined as such. "Consistent state" is a
24 general characteristic of the set of all blocks comprising the data and structure of the file system,
25 as reflected in Sun's construction. NetApp's attempt to narrow the meaning to an on-disk
26 structure rooted by the root inode (i.e., a consistency point) creates confusion and is at odds with
27 the teaching of the specification.
28

III. '715 PATENT

A. “Associating the data blocks with one or more storage blocks across the plurality of stripes as an association,” (Claims 21 and 52), and “The association to associate the data blocks with one or more storage blocks across the plurality of stripes” (Claim 39)

56. U.S. Patent No. 7,200,715 (“the ’715 patent”) discusses a system and method for controlling storage of data over arrays of disk drives. (’715 patent, col. 1:7-9.) In file systems, data is organized into files. A file is a unit of storage that distinguishes one set of information from another. For example, a file can be a document, a sound recording, a movie, or a computer application. In dealing with files most computers systems treat them as logically contiguous sets of data blocks much like a book is a contiguous set of pages. In the system described in the ’715 patent, the data blocks of a file are all fixed in length and are the same size as storage blocks on disk. This allows the system to map each data block to a respective storage block on disk.

57. As described in the ’715 patent and as is well known in the art, multiple disk drives can be used together in parallel to form a “redundant array of inexpensive/independent disks,” called a “RAID” system. (’715 patent, col. 1:27-36.) RAID systems provide improved reliability and/or high data transfer rates relative to single disk drive systems because in a RAID system data can be written across many disk drives in parallel and stored redundantly. (’715 patent, col. 1:32-36.) In some RAID systems, if a disk drive fails, it can be replaced without shutting down the entire system. (’715 patent, col. 1:43-45.)

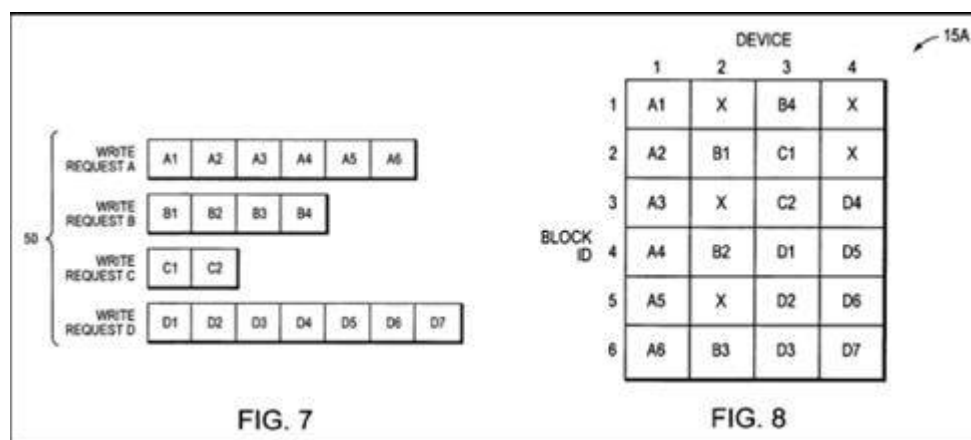
58. In a RAID system, data blocks are written in to “one storage block on each disk drive in an array of drives in the system.” (’715 patent, col. 1:37-39.) Blocks of data are written across the disk drives of a RAID system in the form of “stripes.” The ’715 patent defines a “stripe” as being “one storage block on each disk drive in an array of drives in the system.” (’715 patent, col. 1:37-39.) This definition is consistent with the definition of the term as it is understood by those of ordinary skill in the art.

59. The ’715 patent provides an example of a RAID array including four disk drives, 90A, 90A', 90A" and 92A. (’715 patent, Fig. 1A.) Data blocks are written across the disk drives “as a series of stripes, 94A, 94A', 94A" (generally 94).” (’715 patent, col. 5:8) In practice a single

1 disk drive may have millions of storage blocks with stripes that may be on the order of 5 to 10
2 storage blocks in length, that is, 5 to 10 disk drives wide.

3 60. In some RAID systems, parity information can be used to reconstruct data from a
4 failed disk. For example, if a file contains three data blocks of information, it could be written
5 across stripe 94A with a single data block residing in a respective storage block of each of disks
6 90A, 90A' and 90A". Parity would then be calculated for the stripe and stored in a corresponding
7 storage block of parity disk 92A. If, for instance, disk 90A' failed, the RAID system could still
8 retrieve the file by accessing the corresponding storage blocks of data disks 90A and 90A" and
9 the parity block on disk 92A. The parity information in disk 92A would be used together with the
10 data from disks 90A and 90A" to recreate the data on failed disk 90A'.

11 61. For a number of reasons, the data blocks in a write request or set of write requests
12 might not be contiguously stored in the disk array but rather, scattered about. As part of the
13 claimed invention, the computer system uses a map to represent the one-to-one correspondence
14 between each data block being written and its corresponding storage block on the storage device.
15 The '715 patent's map is called an "association" in the claims and variously called a "map", an
16 "association," and/or "block layout information" in the specification. One example is illustrated
17 in Figure 8, reproduced below:



22 62. Figure 7 illustrates four write requests for data blocks that are part or all of four
23 files, i.e., contiguous data blocks A1 through A6 from File A, blocks B1 through B4 from File B,
24
25
26
27
28

1 *etc.* Figure 8 is a representation of the mapping of the data blocks in Figure 7 to physical drives
2 in a RAID system; compare, for example, Figure 8 with Figure 1A. Particularly, Figure 8 is the
3 map, “an association 15A” (’715 patent, col. 15:35), that pictorially shows the one-to-one
4 correspondence between the data blocks to be written and the storage blocks into which the data
5 blocks will be written. Of course, the internal computer representation of this map will be
6 different than this figure.

7 63. A parity device is not shown in Figure 8. However, under the RAID 4 system
8 described in the ’715 patent, a fifth device (the “parity” device) would be used to store parity
9 information. Parity information would be written to the parity device for each of the stripes (1-6)
10 of the array, one set of parity information generated for each of the six stripes.

11 64. The invention claims to be an improvement over prior arrays of drives that
12 performed individual stripe writes by writing more than one stripe at a time, that is, a “plurality of
13 stripes” (’715 patent, Claim 21, col. 21:47) in a “single write request [*sic*, transaction]” (’715
14 patent, Claim 21, col. 21:51-52). The ’715 patent claims that all prior art systems wrote a single
15 stripe at a time. (’715 patent, col. 9:28-31.) Thus, a key distinction that the patent claims over the
16 prior art is that the one-to-one association described traverses multiple stripes.

17 65. In the context of Figures 7 and 8, and in accordance with the claimed invention,
18 there is only one write transaction to the array including all of the data blocks. A write
19 transaction decision is made when a sufficient quantity of data blocks are available as determined
20 by a number of criteria. (’715 patent, col. 10:55-58). For example, the write transaction might be
21 delayed until an optimal number of storage blocks per device are ready to be written. (’715
22 patent, col. 10:61-2). Thus, if the criterion were four storage blocks per device, then the map of
23 Figure 8 would meet this criterion and there would be one write transaction to the array. In such
24 a write transaction, each drive would write up to six storage blocks, all five devices (devices 1-4
25 and the parity device) writing in parallel. In the context of Figure 8, for example, device 1 would
26 write data blocks A1 thru A6 into its storage blocks, device 3 would write data blocks B4, C1,
27 C2, D1, D2 and D3 into its storage blocks, etc.

28 66. I have reviewed the claims of the ’715 patent, and in my opinion, claims 21, 39

1 and 52 are indefinite because the terms “associating the data blocks with one or more storage
2 blocks across the plurality of stripes as an association” and “the association to associate the data
3 blocks with one or more storage blocks across the plurality of stripes” (the “associating
4 limitations”) are inherently inconsistent, one of ordinary skill in the art could not determine
5 whether a product practices the claims, and the claims fail to capture the key feature that the ’715
6 patent explains makes the invention different from the prior art.

7 67. When the words of the associating limitations are given their full range of plain
8 and ordinary meaning, the claims purport to cover at least two situations: (i) associating the data
9 blocks with a single storage block across multiple stripes as an association; or (ii) associating the
10 data blocks with more than one storage block across a plurality of stripes as an association.
11 Because the claim uses the term “or,” satisfying either situation would meet the limitation.
12 However, because a stripe consists of multiple storage blocks and any one storage block cannot
13 exist across multiple stripes, the first situation states an impossibility. Therefore, in my opinion,
14 one of ordinary skill in the art would not be able to determine whether a particular product
15 satisfies the first situation and thus cannot reasonably determine whether a product infringes the
16 claim.

17 68. NetApp argues that the invention “strictly requires” an association of multiple
18 storage blocks across multiple stripes of an array, which distinguishes the ’715 patent from the
19 prior art (NetApp Response Brief, 35-38). This argument is also made repeatedly throughout the
20 file history where NetApp urged the patent office to allow the claims because the prior art did not
21 teach mapping or associating “each data block with a respective one of the storage blocks across a
22 plurality of stripes.” (See Sun’s Op. Brief, pages 35-37). This critical “requirement”, however, is
23 missing from claims 21, 39 and 52. Instead, claims 21, 39 and 52 were all written to cover what
24 NetApp refers to as a “degenerate” case where the association consists of only a single storage
25 block that cannot exist across multiple stripes. Therefore, the claims as written cover subject
26 matter that does not include a critical feature of the patent and that is not distinguishable from the
27 prior art.

28 69. If the Court were to determine that claims 21, 39 and 52 are not indefinite, in my

1 opinion, the only construction for the terms “associating the data blocks with one or more storage
2 blocks across the plurality of stripes as an association” and “the association to associate the data
3 blocks with one or more storage blocks across the plurality of stripes” that is consistent with the
4 specification and file history is the construction proposed by Sun (i.e., “associating each data
5 block with a respective one of the storage blocks across the plurality of stripes as an association”).
6 The specification repeatedly describes the invention as associating data blocks to storage blocks
7 in a one-to-one correspondence, i.e., each data block is associated with a respective one of the
8 storage blocks. (’715 patent, Abstract, 9:16-19, 13:37-39, 13:20-22, 13:2-5). This one-to-one
9 association is the only type of association described in the ’715 patent. Sun’s proposed
10 construction captures this inherent feature of the association as set forth in the ’715 specification,
11 while NetApp’s proposed construction does not.

12 70. Furthermore, in my opinion, NetApp’s construction, which replaces the claim term
13 “storage blocks” with the more general term “locations” has no basis in the specification, file
14 history, or in the art. Both the specification and the prosecution history clearly define stripes in
15 terms of storage blocks and require a one-to-one mapping of data blocks to storage blocks.

16 71. In my opinion, Sun’s construction is consistent with the disclaimers NetApp made
17 in the prosecution history of the ’715 patent, while NetApp’s construction contradicts its own
18 disclaimers in the prosecution history and therefore cannot be correct.

19 72. Consistent with the teachings of the ’715 patent specification, NetApp argued
20 during prosecution of the ’715 patent that the novel “association” required a one-to-one mapping
21 of data blocks to storage blocks over a plurality of stripes. In at least three different amendments,
22 NetApp urged the Patent Office to allow the claims because the prior art did not teach “mapping”
23 or associating “each data block with a respective one of the storage blocks” across a plurality of
24 stripes. Sun’s Opening Brief at 35-37. NetApp made this argument with respect to all of the
25 claims, including claims 21, 39 and 52. Furthermore, the Examiner agreed and allowed claims
26 21, 39 and 52 for this very reason—i.e., because the prior art allegedly lacked “associating each
27 [of the] data blocks to be stored with a respective one of the storage blocks across the plurality of
28 stripes for a single write operation.” Sun’s Opening Brief at 35-37; see also, May 17, 2005 Office

1 Action at Supp. Williamson Decl., Ex. 1. Sun's proposed construction captures this feature by
2 requiring "associating each data block with a respective one of the storage blocks across the
3 plurality of stripes"—the feature that NetApp argued made the inventions patentable over the
4 prior art. In contrast, NetApp's proposed construction eliminates the requirement of associating
5 each data block with a respective one of the storage blocks, a feature which it repeatedly relied on
6 to overcome the prior art.

7 I declare under the penalty of perjury under the laws of the United States of America that
8 the foregoing is true and correct. This declaration is executed on this August 1, 2008, at East Palo
9 Alto, California.

10
11 
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28

DR. SCOTT BRANDT